

Numerical simulation code for self-gravitating Bose-Einstein condensates

Enikő J. M. Madarassy^a, Viktor T. Toth^b

^a*Division of Astronomy and Space Physics, Uppsala University, 751 20 Uppsala, Sweden*

^b*Ottawa, Ontario, K1N 9H5 CANADA*

Abstract

We completed the development of simulation code that is designed to study the behavior of a conjectured dark matter galactic halo that is in the form of a Bose-Einstein Condensate (BEC). The BEC is described by the Gross-Pitaevskii equation, which can be solved numerically using the Crank-Nicholson method. The gravitational potential, in turn, is described by Poisson's equation, that can be solved using the relaxation method. Our code combines these two methods to study the time evolution of a self-gravitating BEC. The inefficiency of the relaxation method is balanced by the fact that in subsequent time iterations, previously computed values of the gravitational field serve as very good initial estimates. The code is robust (as evidenced by its stability on coarse grids) and efficient enough to simulate the evolution of a system over the course of 10^9 years using a finer $(100 \times 100 \times 100)$ spatial grid, in less than a day of processor time on a contemporary desktop computer.

Keywords:

1. Introduction

The rotation of spiral galaxies does not follow simple predictions based on Kepler's laws. Instead, the rotational velocity curve of most spiral galaxies, plotted as a function of radial distance from the galaxy center, remains "flat" for a broad range of radii. The standard proposal to resolve this problem is to presume the existence of a "dark matter halo", which contains most of the mass of a spiral galaxy. To maintain consistency with the predictions of the most broadly accepted cosmological models, this halo must necessarily consist of "exotic" matter, i.e., matter predominantly composed of something

other than baryons. The halo must also be collisionless and not interacting with baryonic matter [1].

The existence of such a halo with a suitable geometry can account for the observed rotation curves of visible matter. However, a difficult problem is to construct a dark matter halo that is gravitationally stable and does not predict excessive dark matter densities in the inner parts of the galaxy where most visible matter resides. This issue is known as the “cuspy halo problem” in the relevant literature [2].

A recent proposal [3, 4, 5, 6, 7] addresses the cusp problem by a dark matter halo that forms a Bose-Einstein condensate (BEC) [8, 9]. The dynamics of a BEC halo may be determined by the balance of the attractive force of gravity and a repulsive effective long-range interaction [10, 11, 12, 13] (see also [14]). As the dark matter halo dominates the gravitational field of a spiral galaxy, a simulation that is restricted to just the halo should be sufficient to determine if a field can be obtained that yields the desired circular orbital velocities.

In the present paper, we discuss a simulation tool that we constructed to explore the dynamics of a galactic BEC halo. Our work is based primarily on our previous simulation of BEC in laboratory conditions, described by the non-linear Schrödinger equation, also known in the literature as the Gross-Pitaevskii equation. Whereas in the laboratory, a BEC characterized by a repulsive interaction is held together by an artificially introduced trapping potential, in the case of a galaxy floating in empty space, the trapping potential must be replaced by self-gravity. A numerical solution must, therefore, simultaneously address the initial value problem of the Gross-Pitaevskii equation and the boundary condition problem of Poisson’s equation.

In Sec. 2, we introduce the dimensionless form of the Gross-Pitaevskii equation used in our computations, and the method used to solve this equation efficiently. In Sec. 3 we discuss Poisson’s equation for gravity and the relaxation method. In Sec. 4 we elaborate on the use of physical units that are suitable for such a simulation in an astrophysical context. The problem of using suitable initial conditions to form a stable halo is briefly discussed in Sec. 5. In Sec. 6 we discuss the implementation of our method in FORTRAN and C++, and also comment on the possible use of GPUs for accelerated computation. The actual programs are summarized in Sec. 7. Finally, our conclusions and outlook are presented in Sec. 8.

2. Solving the Gross-Pitaevskii equation

A self-interacting, optionally rotating Bose-Einstein condensate is described accurately by a form of the time-dependent nonlinear Schrödinger equation known as the Gross-Pitaevskii equation [15, 16]. For computational purposes, it is advantageous to use a dimensionless form of this equation, which takes the form [17]:

$$(i - \gamma) \frac{\partial \psi}{\partial t} = \hat{H} \psi, \quad (1)$$

where γ is a softening parameter ($\gamma = 0$ is a valid choice), ψ is the wave function, t is time, and \hat{H} is the Hamilton-operator, which in turn is given by

$$\hat{H} = -\frac{1}{2} \nabla^2 + V. \quad (2)$$

The potential V is the sum of classical potentials (e.g., gravitational potential, trapping potential), the chemical potential, the non-linear term, and a rotational term:

$$V = \phi + \kappa |\psi|^2 - \mu - \Omega L_z, \quad (3)$$

where κ represents the interaction strength, Ω is the angular velocity, and $L_z = i(x\partial_y - y\partial_x)$. We assume that the condensate's net rotation is in the $x - y$ plane.

In earlier work [18, 19, 20, 21], we solved the Gross-Pitaevskii equation can be solved numerically using the Crank-Nicholson method in combination with Cayley's formula [22], in the presence of an isotropic trapping potential (for a numerical solution in the presence of an anisotropic trap, see [23, 24]). In particular, the use of Cayley's formula ensures that the numerical solution remains stable and the unitarity of the wavefunction is maintained.

The value ψ_{n+1} of the wavefunction at the $(n+1)$ -th time step is obtained from the known values ψ_n at the n -th time step by solving the following equation:

$$\left(1 + \frac{1}{2} i \Delta t \hat{H}\right) \psi^{n+1} = \left(1 - \frac{1}{2} i \Delta t \hat{H}\right) \psi^n. \quad (4)$$

After evaluating the right-hand side given ψ_n , the left-hand side can be solved for. If \hat{H} is a linear operator, this is a linear system of equations for the unknown values ψ_{n+1} .

In the one-dimensional case, the Hamilton operator reads

$$\hat{H} = -\frac{1}{2} \frac{\partial^2}{\partial x^2} + V. \quad (5)$$

The second derivative can be approximated as a finite difference:

$$\frac{\partial^2 \psi}{\partial x^2} = \frac{\psi_{k-1} - 2\psi_k + \psi_{k+1}}{(\Delta x)^2}, \quad (6)$$

Substituting this into Eq. (4), we obtain

$$\begin{aligned} & \left[1 - \frac{i\Delta t}{2} \left(V + \frac{1}{(\Delta x)^2} \right) \right] \psi_k^{n+1} - \frac{i\Delta t}{4(\Delta x)^2} (\psi_{k-1}^{n+1} + \psi_{k+1}^{n+1}) \\ & = \left[1 - \frac{i\Delta t}{2} \left(V + \frac{1}{(\Delta x)^2} \right) \right] \psi_k^n + \frac{i\Delta t}{4(\Delta x)^2} (\psi_{k-1}^n + \psi_{k+1}^n). \end{aligned} \quad (7)$$

This is a linear system of equations for the values of ψ^{n+1} on the left-hand side, if the values of ψ^n on the right-hand side are known. Moreover, the form of this system of equations is tridiagonal, which can be solved highly efficiently using the Thomas algorithm [22].

In the three-dimensional case, one could proceed with solving for ψ^{n+1} directly, but as the system of equations is no longer tridiagonal, the efficiency related to tridiagonal systems is lost. This is why it is preferable to use the *alternating-direction implicit method*, calculating the one-dimensional solution in the x , y and z directions, using $\frac{1}{3}\Delta t$ for the time step and $\frac{1}{3}V$ for the potential. This approach is possible because the Hamilton operator can be viewed as a sum of three operators, $\hat{H} = \hat{H}_x + \hat{H}_y + \hat{H}_z$ allowing us to solve numerically using fractional time steps as follows:

$$\begin{aligned} \hat{H}_x &= -\frac{1}{2} \frac{\partial^2}{\partial x^2} + \frac{1}{3}V, \\ \hat{H}_y &= -\frac{1}{2} \frac{\partial^2}{\partial y^2} + \frac{1}{3}V, \\ \hat{H}_z &= -\frac{1}{2} \frac{\partial^2}{\partial z^2} + \frac{1}{3}V. \end{aligned} \quad (8)$$

Substituting these into Eq. (7) (that is, replacing V with $V/3$ and Δx , respectively, with Δx , Δy or Δz), using a time step of $\Delta t/3$ we obtain the

three fractional iteration steps that correspond to a full iteration with time step Δt .

As to the nonlinear term, it can be dealt with by a simple iteration that converges rapidly. Specifically, we calculate the non-linear term on the left-hand side by substituting ϕ^n in place of ϕ^{n+1} and solve the system of equations; we then use this solution to recalculate the non-linear term and solve again, until convergence is obtained. Because in our case, the non-linear term is a cubic term, convergence is very rapid.

3. Solving Poisson's equation

The (non-relativistic) gravitational field corresponding to a distribution of matter characterized by density ρ is given by Poisson's equation for gravity:

$$\nabla^2 \phi = 4\pi G \rho, \quad (9)$$

where G is the gravitational constant. For a BEC, the mass density is given by $\rho = |\psi|^2 m$, where m is the mass of the BEC particle. When the BEC condensate is described using the dimensionless Gross-Pitaevskii equation, $m = 1$.

A moderately efficient numerical method for solving Poisson's equation is the *relaxation method*. This method is based on the finite differences approximation of the second derivative in Poisson's equation:

$$\begin{aligned} \nabla^2 \phi &= \frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} + \frac{\partial^2 \phi}{\partial z^2} \\ &= \frac{\phi(x - \Delta x, y, z) - 2\phi(x, y, z) + \phi(x + \Delta x, y, z)}{\Delta x^2} \\ &\quad + \frac{\phi(x, y - \Delta y, z) - 2\phi(x, y, z) + \phi(x, y + \Delta y, z)}{\Delta y^2} \\ &\quad + \frac{\phi(x, y, z - \Delta z) - 2\phi(x, y, z) + \phi(x, y, z + \Delta z)}{\Delta z^2} + \mathcal{O}(\Delta^2), \end{aligned} \quad (10)$$

where Δ^2 is the largest of Δx^2 , Δy^2 and Δz^2 . If the left-hand side of this equation is given (e.g., by Poisson's equation), this equation can be solved for $\phi(x, y, z)$. The essence of the relaxation method is the realization that these solutions provide successively refined approximations of $\phi(x, y, z)$. In

other words, we obtain the iteration formula

$$\begin{aligned}
\phi_{k+1}(x, y, z) = & \frac{\Delta x^2 \Delta y^2 \Delta z^2}{2(\Delta x^2 \Delta y^2 + \Delta y^2 \Delta z^2 + \Delta z^2 \Delta x^2)} \\
& \left[\frac{\phi_k(x - \Delta x, y, z) + \phi_k(x + \Delta x, y, z)}{\Delta x^2} \right. \\
& + \frac{\phi_k(x, y - \Delta y, z) + \phi_k(x, y + \Delta y, z)}{\Delta y^2} \\
& \left. + \frac{\phi_k(x, y, z - \Delta z) + \phi_k(x, y, z + \Delta z)}{\Delta z^2} - 4\pi G |\psi|^2 \right]. \quad (11)
\end{aligned}$$

This method is accurate but its convergence is slow. However, after the initial configuration of ϕ is determined and we iterate the system to the next timestep using the Gross-Pitaevskii equation, ψ and, consequently, ρ will change very little. Therefore, using the values of ϕ^n at timestep n as the initial estimate for ϕ^{n+1} at timestep $(n+1)$, very rapid convergence is often obtained after just a few iterations.

4. Choice of units

In order to put the code presented in this paper to use in an astrophysical context, it is necessary to restore dimensional units.

Use of the dimensionless form of the Gross-Pitaevskii equation amounts to choosing units such that $\hbar = 1$ and also $m = 1$, where m is the mass of the BEC particle. Given units of length [L], mass [M] and time [T], the choice of $\hbar = 1$ amounts to

$$1 \frac{([L] \text{ m})^2 \cdot ([M] \text{ kg})}{([T] \text{ s})} = 10^{-34} \frac{\text{m}^2 \cdot \text{kg}}{\text{s}}. \quad (12)$$

Conversely, choosing a specific numerical value for G amounts to making the choice

$$G \frac{([L] \text{ m})^3}{([M] \text{ kg}) \cdot ([T] \text{ s})^2} = 6.67 \times 10^{-11} \frac{\text{m}^3}{\text{kg} \cdot \text{s}^2}. \quad (13)$$

After choosing a specific value for [L], these two equations allow us to determine [M] and [T]:

$$[T] = 5.3 \times 10^{14} [L]^{5/3} \sqrt[3]{G} \text{ s}, \quad (14)$$

$$[M] = 5.3 \times 10^{-20} [L]^{-1/3} \sqrt[3]{G} \text{ kg}. \quad (15)$$

Specifically, if we choose our unit of length to be $1 \text{ kpc} \simeq 3 \times 10^{19} \text{ m}$, we get

$$[\text{T}] = 1.5 \times 10^{47} \sqrt[3]{G} \text{ s}, \quad (16)$$

$$[\text{M}] = 1.7 \times 10^{-26} \sqrt[3]{G} \text{ kg} \simeq 10 \sqrt[3]{G} \text{ GeV}. \quad (17)$$

Choosing $G = 10^{-100}$ yields the units

$$[\text{T}] = 7 \times 10^{13} \text{ s} \simeq 2.2 \times 10^6 \text{ year}, \quad (18)$$

$$[\text{M}] = 8 \times 10^{-60} \text{ kg}, \quad (19)$$

corresponding to a BEC particle mass of $4.4 \times 10^{-24} \text{ eV}$.

Finally, velocity is measured in units of $\text{kpc}/(2.2 \text{ million years}) \simeq 430 \text{ km/s}$.

5. Initial and boundary conditions

Numerically solving a system of coupled differential equations requires a set of initial and boundary conditions.

Specifically, solving the Gross-Pitaevskii equation requires an initial field configuration $\psi(x, y, z)$ at $t = 0$. In turn, the numerical solution of Poisson's equation for gravity needs boundary conditions in the form of values $\phi(x_{\min}, y, z)$, $\phi(x_{\max}, y, z)$, $\phi(x, y_{\min}, z)$, $\phi(x, y_{\max}, z)$, $\phi(x, y, z_{\min})$ and $\phi(x, y, z_{\max})$.

For the purpose of testing our code, we chose a very simple initial density profile, related to solutions of the Lane-Emden equation, centered around the origin at $x = 0, y = 0, z = 0$:

$$\rho = \rho_0(R^2 - r^2), \quad (20)$$

where $r^2 = x^2 + y^2 + z^2$ and R is a characteristic radius. Given ρ , we calculate $\psi = \sqrt{\rho}$, making the initial value of ψ purely real everywhere.

We emphasize that this choice does not necessarily represent a physically viable configuration; it was strictly used for code testing and validation. Application of the code involves, among other things, choosing initial density profiles that reflect valid physical assumptions. For instance, one may opt to use a Navarro-Frank-White distribution [25] to model the initial halo density at $t = 0$.

As to the boundary condition, we simply assume that the gravitational field vanishes on the boundaries of the simulation volume. Since in the context of Newtonian gravity, the gravitational field is indeterminate up to an

additive constant, this amounts to the assumption the gravitational field is constant on the boundary. While this is somewhat artificial, one may justify this choice by noting that real galaxies exist in an external gravitational field that in turn is determined by other, more distant galaxies and clusters; this external field is imposed upon the dynamics of a real, physical galaxy the same way we impose our boundary condition on the model galaxy. In any case, if the simulation volume is sufficiently large compared to the galaxy being simulated, the geometry of the boundary will play no significant role in the model galaxy’s evolution.

6. Implementation and testing

Our code was originally implemented in FORTRAN 90. Later, however, we decided to port the code to C++, which resulted in a twofold performance improvement.

We also experimented with a GPGPU¹ accelerated version that was designed to run using the OPENCL library, on an AMD 6900-class graphics card. This version yielded considerable improvement in performance, but only when single-precision arithmetic was used. The magnitude of the quantities needed in the astrophysical context (e.g., $G = 10^{-100}$) precluded the use of single-precision floats. The performance improvement of the GPGPU version was not sufficient to justify further rewriting the code; therefore, the GPGPU implementation was, for the time being, abandoned, although the code remains functional.

The C++ version of the code was heavily tested using various grid sizes. When testing the code, we took to heart the important advice offered by the authors of Ref. [22]: “*You should always first run your programs on very small grids, e.g., 8×8 , even though the resulting accuracy is [...] poor [...] [N]ew instabilities sometimes do show up on larger grids, but old instabilities never (in our experience) just go away.*” We determined that our code runs very well on a grid size of $20 \times 20 \times 20$, and the simulation is very rapid; this makes it easy to select physically interesting test cases that can be further analyzed using a much finer grid. On the other hand, we found that even with a grid size of $120 \times 120 \times 120$, a simulation that models the evolution over 10^9 years can be completed in the course of a day or so on modern desktop computer hardware.

¹General Purpose Graphics Processing Unit

We specifically tested the numerical robustness of our code by verifying that the wavefunction ψ remains unitary. Even after 50,000 iterations, $\int_V |\psi|^2 dV$, when evaluated for the entire simulation volume, remained within a few percent of its initial value.

Specifically, we ran a test simulation in a $100 \times 100 \times 100$ kpc³ volume, using an $80 \times 80 \times 80$ cells, using a time step of $\Delta t = 0.01$ units of time (corresponding to $\sim 22,000$ years.) For the entire volume, $\int |\psi|^2 dV = 1.7 \times 10^{102}$, corresponding to a total BEC mass of $\sim 6.8 \times 10^{12} M_\odot$. The total simulation time of 50,000 iterations corresponds to $\sim 1.1 \times 10^9$ years; the results of this simulation are shown in Fig. 1.

7. Program summary

Our self-gravitating BEC code has several components. A stand-alone executable compiled from FORTRAN (`bec3p.f90`) or C++ (`bec3p.c++`) source performs the simulation. User-definable parameters are in header files (`parameters3.f90` or `parameters3.h`). A change in parameters necessitates recompiling the executables, but this is quickly accomplished using the supplied make file (`Makefile`). The simulation code creates data files at set intervals, specifically formatted to facilitate plotting using the `gnuplot` utility. A UNIX style shell script (`animate.sh`) is provided that uses `gnuplot` to create animations in the form of animated GIF files. Finally, yet another shell script (`video.sh`) can be used to convert these GIF files into MPEG-4 video animations.

The structure of the main source file (`bec3p.f90` or `bec3p.c++`) is as follows:

- The main program (or `_tmain` function in the C++ implementation) initializes parameters and executes the main simulation loop.
- The function `init` creates the initial distribution. This is the only user-modifiable subroutine in the main program file; in the test implementation, it creates a naive initial distribution inspired by the Lane-Emden equation.
- The function `get_U` computes the potential term (without rotation) in the Gross-Pitaevskii equation, as given in Eq. (3).
- The functions `calc_rhs_x`, `calc_rhs_y` and `calc_rhs_z` compute the right-hand side of the Gross-Pitaevskii equation (1).

- The functions `solve_x`, `solve_y` and `solve_z` solve for the left-hand side of the Gross-Pitaevskii equation according to the scheme represented by Eq. (4).
- The function `thomas` solves a tridiagonal system of linear equations using the Thomas algorithm.
- The function `get_normsimp` obtains a numerical approximation of $\int_V |\psi|^2 dV$ using Simpson's formula.
- The functions `movieX`, `movieY` and `movieZ` generate output after a set number of iterations (controlled by the variable `nstep2`) that is set in `parameters3.h` or `parameters3.f90` in a format that is suitable for use with `gnuplot`.
- The function `get_phi` computes the gravitational potential corresponding to the current values of ψ using the relaxation method.
- The function `get_density` computes the density $|\psi|^2$.

Output files produced by the program include the following:

- The files `densZnnnnnnnn.dat` contain values of density ($|\psi|^2$), sorted and grouped to facilitate plotting with `gnuplot` in the plane perpendicular to the Z -axis (i.e., the XY plane). Similarly, the files `densXnnnnnnnn.dat` and `densYnnnnnnnn.dat` contain density values sorted and grouped for plotting in the plane perpendicular to the X and Y -axis.
- The files `gravZnnnnnnnn.dat`, `gravXnnnnnnnn.dat` and `gravYnnnnnnnn.dat` contain values of the gravitational potential, arranged similarly.
- The files `phasZnnnnnnnn.dat`, `phasXnnnnnnnn.dat` and `phasYnnnnnnnn.dat` contain values of the complex phase of ψ , again arranged as before.

The shell script `animate.sh` provides simple visualization. By default, this script create an animated GIF of density values, in the plane perpendicular to the Z -axis and across the origin, with the color range normalized to the 1000th iteration step. Running `animate.sh -h` prints a brief help message that lists all the options of this script; in particular, the option `-p` that allows the user to change the plane for visualization and the option `-t` that allows

the user to change the plot type. The script can also output an animated plot of the estimated Keplerian rotational velocity (option `-t vrot`), for circular orbits centered around the origin in the XY plane.

Finally, the shell script `video.sh` can be used to convert animated GIF output into MPEG-4 video. For this script to work, the utilities `ffmpeg` and `mplayer` must be installed on the workstation where the script is being used.

8. Conclusions

Development of the software code described in this paper is now complete, with the code yielding expected results for test cases. The next step is to find suitable initial conditions to model a BEC dark matter halo that may surround a real galaxy. The question is whether halo configurations can be found that are stable over time scales of 10^{10} years, and yield circular orbital velocities that remain approximately constant at different radii.

Results of this on-going investigation will be reported when they become available.

Acknowledgements

EJMM thanks Profs. C. F. Barenghi and M. Tsubota for help with the development of the first version of the code presented in this manuscript.

References

- [1] S. Weinberg, *Cosmology*, Oxford University Press, 2008.
- [2] W. J. G. de Blok, The Core-Cusp Problem, *Advances in Astronomy* 2010 (2010).
- [3] S.-J. Sin, Late-time phase transition and the galactic halo as a Bose liquid, *Phys. Rev. D* 50 (1994) 3650–3654.
- [4] S. U. Ji, S. J. Sin, Late-time phase transition and the galactic halo as a Bose liquid. II. The effect of visible matter, *Phys. Rev. D* 50 (1994) 3655–3659.
- [5] W. Hu, R. Barkana, A. Gruzinov, Fuzzy Cold Dark Matter: The Wave Properties of Ultralight Particles, *Physical Review Letters* 85 (2000) 1158–1161.

- [6] V. Sahni, L. Wang, New cosmological model of quintessence and dark matter, *Phys. Rev. D* 62 (2000) 103517.
- [7] C. G. Böhrer, T. Harko, Can dark matter be a Bose Einstein condensate?, *J. Cosmol. Astropart. Phys.* 6 (2007) 25–+.
- [8] S. N. Bose, Plancks Gesetz und Lichtquantenhypothese, *Zeitschrift für Physik* 26 (1924) 178–181.
- [9] A. Einstein, Quantentheorie des einatomigen idealen gases: Zweite abhandlung, *Sitzungber. Kgl. Akad. Wiss.* (1925).
- [10] M. I. Khlopov, B. A. Malomed, I. B. Zeldovich, Gravitational instability of scalar fields and formation of primordial black holes, *Mon. Not. R. Astron. Soc.* 215 (1985) 575–589.
- [11] I. Dymnikova, L. Koziel, M. Khlopov, S. Rubin, Quasilumps from First Order Phase Transitions, *Gravitation and Cosmology* 6 (2000) 311–318.
- [12] M. Y. Khlopov, S. G. Rubin, A. S. Sakharov, Strong Primordial Inhomogeneities and Galaxy Formation, *ArXiv Astrophysics e-prints* (2002).
- [13] M. Y. Khlopov, S. G. Rubin, A. S. Sakharov, Primordial structure of massive black hole clusters, *Astroparticle Physics* 23 (2005) 265–277.
- [14] M. Khlopov, S. Rubin, *Cosmological Pattern Of Microphysics In The Inflationary Universe*, *Fundamental Theories of Physics*, Springer, 2004.
- [15] E. Gross, Structure of a quantized vortex in boson systems, *Il Nuovo Cimento* (1955-1965) 20 (1961) 454–477. 10.1007/BF02731494.
- [16] L. P. Pitaevsk, Vortex lines in an imperfect bose gas, *Soviet Physics JETP-USSR* 13 (1961).
- [17] K. Kasamatsu, M. Tsubota, Dynamical Vortex Phases in a Bose-Einstein Condensate Driven by a Rotating Optical Lattice, *Physical Review Letters* 97 (2006) 240404.
- [18] E. J. M. Madarassy, C. F. Barenghi, Vortex Dynamics in Trapped Bose-Einstein Condensate, *Journal of Low Temperature Physics* 152 (2008) 122–135.

- [19] E. Madarassy, C. Barenghi, Disordered vortex arrays in a two-dimensional condensate, *Geophysical and Astrophysical Fluid Dynamics* 103 (2009) 269–278.
- [20] E. J. M. Madarassy, A method to create disordered vortex arrays in atomic Bose-Einstein condensates, *Canadian Journal of Physics* 87 (2009) 1013–1019.
- [21] E. J. M. Madarassy, Decay of soliton-like perturbations into vortex-anti vortex pairs, *Rom. Journ. Phys.* 55 (2010) 249–258.
- [22] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, *Numerical Recipes in C, Second Edition*, Cambridge University Press, 1992.
- [23] P. Muruganandam, S. K. Adhikari, Fortran programs for the time-dependent Gross-Pitaevskii equation in a fully anisotropic trap, *Computer Physics Communications* 180 (2009) 1888–1912.
- [24] D. Vudragović, I. Vidanović, A. Balaž, P. Muruganandam, S. K. Adhikari, C programs for solving the time-dependent Gross-Pitaevskii equation in a fully anisotropic trap, *Computer Physics Communications* 183 (2012) 2021–2025.
- [25] J. F. Navarro, C. S. Frenk, S. D. M. White, A Universal Density Profile from Hierarchical Clustering, *Astrophys. J.* 490 (1997) 493–+.

Figure 1: Illustrative example of the evolution of a rotating, self-gravitating BEC, in a $100 \times 100 \times 100 \text{ kpc}^3$ volume, simulated using an $80 \times 80 \times 80$ spatial grid. The calculated rotational velocity and density are shown after the 1, 250 and 500 iteration time units, the latter corresponding to $\sim 1 \text{ Gyr}$. The total mass of this condensate is $\sim 6.8 \times 10^{12} M_\odot$.

